
Logical Foundations and Complexity of 4QL, a Query Language with Unrestricted Negation

Jan Małuszyński^a and Andrzej Szalas^{a,b}

^a*Department of Computer and Information Science, University of Linköping,
SE-581 83 Linköping, Sweden, email: jan.maluszynski@ida.liu.se.*

^b*Institute of Informatics, Warsaw University, Banacha 2, 02-097 Warsaw, Poland
email: andrzej.szalas@mimuw.edu.pl*

ABSTRACT. *The paper discusses properties of a DATALOG[¬]-like query language 4QL, originally outlined by Małuszyński and Szalas [MS11]. 4QL allows one to use rules with negation in heads and bodies of rules. It is based on a simple and intuitive semantics and provides uniform tools for “lightweight” versions of known forms of nonmonotonic reasoning. Negated literals in heads of rules may naturally lead to inconsistencies. On the other hand, rules do not have to attach meaning to some literals. Therefore 4QL is founded on a four-valued semantics, employing the logic introduced in [MSV08, VMS09] with truth values: ‘true’, ‘false’, ‘inconsistent’ and ‘unknown’. In addition, 4QL is tractable w.r.t. data complexity and captures PTIME queries. Even though DATALOG[¬] is known as a concept for the last 30 years, to our best knowledge no existing approach enjoys these properties.*

In the current paper we:

- *investigate properties of well-supported models of 4QL*
- *prove the correctness of the algorithm for computing well-supported models*
- *show that 4QL has PTIME data complexity and captures PTIME.*

KEYWORDS: *query language, data complexity, paraconsistent semantics, DATALOG[¬]*

1. Introduction and Preliminaries

This is a companion paper to our conference paper [MS11]. It is motivated by the problem of handling explicit negative knowledge in a rule language. This means that negation can appear in bodies and heads of rules, as done in DATALOG[¬] (see, e.g., [AHV96]). However, in contrast to the traditional query languages based on nonmonotonic logics initially derived from the Closed World Assumption [Rei78, AD98] (CWA) we focus our attention on explicit negative knowledge. In [MS11] we outlined a novel lightweight approach to this problem, separating the issues of incomplete/inconsistent knowledge and nonmonotonicity. To make separation between monotonic-nonmonotonic clear, we first follow the Open World Assumption (OWA) in the presence of the explicit negation and later provide simple con-

structs allowing one to structure knowledge bases in the form of modules and to query such modules. As shown in [MS11], these constructs provide a natural way to express CWA and “lightweight” versions of other known forms of nonmonotonic reasoning.

These methodological assumptions led to a DATALOG[¬]-like query language, 4QL, founded on a four-valued semantics. The semantics employs the logic introduced in [MSV08, VMS09] with truth values: ‘true’, ‘false’, ‘inconsistent’ and ‘unknown’, further denoted by t , f , i and u , respectively.

For the four-valued query language 4QL proposed in [MS11] we defined a semantics based on the notion of well-supported models and we proposed an algorithm for computing such models. The language:

- allows one to use rules with unrestricted negation (in heads and bodies)
- has a simple and intuitive semantics
- allows for “lightweight” versions of known forms of nonmonotonic reasoning.

Due to page limits, [MS11] does not contain important properties of well-supported models, nor proofs of theorems and complexity analysis. In the current paper we therefore:

- investigate properties of well-supported models of 4QL
- prove the correctness of the algorithm for computing well-supported models
- show that 4QL has PTIME data complexity and captures PTIME.

Even though DATALOG[¬] is known as a concept for 30 years, to our best knowledge no existing approach enjoys these properties.

In Section 2 we discuss related work. Next, in Section 3, we provide syntax and semantics of 4QL via the notion of well-supported models and continue in Section 4 with properties of well-supported models. Section 5 is devoted to an algorithm for computing well-supported models and a proof of its correctness. In Section 6 we extend basic 4QL by very simple constructs which allow us to show in Section 7 that the extended 4QL captures PTIME on ordered databases. Finally, Section 8 concludes the paper.

2. Related Work

While DATALOG[¬] has attracted much attention (see, e.g., [AHV96] and references therein), a tractable and at the same time intuitive semantics for DATALOG[¬] is not reported in the literature. The problem is often addressed by taking DATALOG with negation or extended logic programs [GL91] as the starting point for paraconsistent extensions, see e.g. [ADP05, dAP07, Ari02, DP98, SI95].¹

Let us now discuss approaches [ADP05, Ari02, dAP07, Fit02] which are closest to ours.

The paper [ADP05] provides a framework for paraconsistent logic programming generalizing some previous work of its authors to an arbitrary complete bilattice of truth-values, where belief and doubt are explicitly represented. In contrast to our approach, where default

1. There is a rich literature on paraconsistent logics (see, e.g., [BCG07] and references therein, in particular [CMM07]).

negation is not used, the focus of this work is semantical integration of explicit and default negation.

The paper [Ari02] proposes a framework for dealing with logic programs with two kinds of negation: an explicit negation and a negation-by-failure. We use a single negation, without referring to provability, as the negation-as-failure does. We find this an advantage of our approach. Also our implication is different. As [Ari02] concentrates on logic programs, database related topics are not considered there. In particular no complexity results concerning finite domains are provided.

Another approach, P-Datalog [dAP07], provides a paraconsistent language for knowledge base integration based on a four-valued logic with the truth ordering which coincides with our truth ordering. However, there are several important differences. First, our language only allows explicit negation (in the head and in the bodies of the rules) while P-Datalog programs only allow negation by default \sim in the rule bodies and use the CWA. Information integration done in P-Datalog uses the truth ordering, while we provide a more general constructs for achieving this goal (extended literals, defined in Section 6). Finally, the rules are interpreted by substantially different implications.

A four-valued semantics for logic programs with negation was also proposed by Fitting (see e.g. [Fit02]). The major difference is that Fitting's semantics is based on the Belnap's logic. This contrasts with our approach since we use a different truth ordering. Second, the semantics of Fitting allows one to derive conclusions from false premises, which we do not allow. Unlike in our framework, a rule of a Fitting program is satisfied if and only if truth values assigned to the head and to the body are equal. Last and not least, the language considered by Fitting does not admit negation in rule heads.

A major difficulty in defining 4QL was handling of disjunction in rule bodies. This paper presents a solution to this problem. A preliminary idea of a four-valued rule language not admitting disjunction in rule bodies appeared in our previous work [VMS09].

Also in the field of the Semantic Web there have been proposals addressing the problem of inconsistent data by defining paraconsistent extensions of Description Logics underlying the Semantic Web Ontology Language OWL2, see, e.g., [MS96, Str97, OW08, MH09, Mai10] (where the issue of nonmonotonicity is not addressed) or by providing rule languages based on defeasible reasoning [Nut94].

3. Syntax and semantics of 4QL

3.1. Syntax of 4QL

For a negative literal ℓ the notation $\neg\ell$ denotes its positive counterpart. We treat propositions as zero-argument relation symbols.

In the sequel, for simplicity, we consider ground rules only and assume that for each head ℓ there is only one rule having the form:

$$\ell := (b_{11}, \dots, b_{1i_1}) \vee (b_{21}, \dots, b_{2i_2}) \vee \dots \vee (b_{m1}, \dots, b_{mi_m}). \quad (1)$$

The rationale for this syntax is as follows. In two-valued logic conjunction of the classical implications of the form $a \rightarrow \ell$ and $b \rightarrow \ell$ is logically equivalent to the implication $a \vee b \rightarrow \ell$. Thus, multiple DATALOG rules with the same rule heads express the disjunction of bodies of these rules. In logic programming this is sometimes used to avoid explicit disjunction in rule bodies. In our four-valued logic the equivalence does not hold (see Remark 4). Therefore, in order to have disjunction in the rule body we have to express it explicitly therein.

In 4QL one can consider admission of multiple rules with the same rule heads, interpreted as four-valued conjunction of such rules. However, intuitive understanding of such four-valued rules is rather difficult. Intuitively we consider a rule with head ℓ as a separate information source concerning ℓ . We prefer to leave to the user how information from different sources is to be combined rather than to impose four-valued conjunction as the only way of combination. Therefore at the basic level of 4QL we do not admit multiple rules with the same head. In the sequel we extend 4QL with a flexible mechanism for fusing information from different sources.

The empty body of a rule is denoted by \emptyset . We sometimes refer to *facts* as to rules of the form $\ell :- \emptyset$.

DEFINITION 1. — *Let ϱ be a rule of the form (1). Then:*

- $rule(\ell) \stackrel{\text{def}}{=} \varrho$
- $head(\varrho) \stackrel{\text{def}}{=} \ell$
- $body(\varrho) \stackrel{\text{def}}{=} (b_{11}, \dots, b_{1i_1}) \vee (b_{21}, \dots, b_{2i_2}) \vee \dots \vee (b_{m1}, \dots, b_{mi_m})$
- for $1 \leq j \leq m$, $\beta_j(\varrho) \stackrel{\text{def}}{=} b_{j1}, \dots, b_{ji_j}$.

Writing $rule(l)$, we always assume that the respective rule with ℓ as its head indeed exists.

REMARK 2. — In the paper we present the case of ground rules only. However, typical rules with variables are allowed, too. We assume that whenever there is a variable appearing in the body of a rule but not in its head then it is assumed to be existentially quantified in its body. For example,

$$p(x, y) :- q(x, y, z).$$

is understood as “ $p(x, y) :- \exists z[q(x, y, z)]$ ”. The existential quantifier is then understood as the disjunction $q(x, y, a_1) \vee \dots \vee q(x, y, a_k)$, where a_1, \dots, a_k are all constants appearing in the database. \square

3.2. The Underlying Logic

Addressing the semantics of DATALOG^{□□} we have made certain methodological choices. First of all, we decided to start with a fully monotonic query language. Therefore, rather than starting with CWA, as most approaches do, we have accepted OWA. This, in turn, naturally introduces \mathbf{u} as a truth value. For example, having just one rule $p :- q$ we do not force p and q to be \mathbf{f} , so have to assign them the value \mathbf{u} . Next, since we allow negated literals in heads of rules, certain conclusions may contradict each other. This also gives rise to a non-classical truth value. Could this value be \mathbf{u} ? If one does not want to distinguish between the lack of

information and inconsistency, such an identification could do the job (see, e.g., [dACM02, CMdA00, DLSS06, DMS06, Fit02, Fit85, NS10]). However, we find it more natural and informative to distinguish between u and i . We then adopt the four-valued logic introduced in [MSV08] as the semantical foundation. The truth tables for the connectives are shown in Table 1.

Table 1. Truth tables for \wedge , \vee , \rightarrow and \neg .

\wedge	f	u	i	t	\vee	f	u	i	t	\rightarrow	f	u	i	t	\neg
f	f	f	f	f	f	f	u	i	t	f	t	t	t	t	f
u	f	u	u	u	u	u	u	i	t	u	t	t	t	t	u
i	f	u	i	i	i	i	i	i	t	i	f	f	t	f	i
t	f	u	i	t	t	t	t	t	t	t	f	f	t	t	t

Notice that our logic is different from the commonly used four-valued Belnap’s logic [Bel77]. In Belnap’s logic two orderings on truth values are considered, known as *knowledge ordering* and *truth ordering*. As shown, e.g., in [Dub08] as well as in our previous work [VMS09, NS10], Belnap’s truth ordering is problematic in areas we focus on and we introduced instead a different truth ordering $f < u < i < t$, which was also independently proposed in [dAP07]. The truth tables for conjunction \wedge and disjunction \vee are respectively defined as minimum and maximum w.r.t. our truth ordering.

REMARK 3. — To motivate our truth ordering, where in particular $u < i$, first note that i contains an evidence that a given proposition is t so in this respect it is “closer” to truth than u . On the other hand, one can argue that i reflects some evidence that the corresponding assertion is not true, while u does not. Observe however, that in the monotonic layer we do not want to derive conclusions from unknown premises, since such derivations lead to nonmonotonicity. In contrary, we want to derive conclusions from inconsistent premises to “correct” heads of rules. To illustrate this point consider the following rule:

$$\text{reduce_temperature} :- \text{high_temperature}.$$

If, in a given interpretation, *high_temperature* is u then one does not want to be forced to derive any conclusions whether to reduce temperature or not, unless nonmonotonic reasoning is used (which is addressed in Section 6).

Consider now a situation, when at some point *high_temperature* is t . Then we also want to conclude that *reduce_temperature* is t . If the situation changes and *high_temperature* becomes i , e.g., by deriving new facts then the previous conclusion as to the truth of *reduce_temperature* is no longer justified and is to be adjusted to i .

The above case indicates that i behaves like t (so can be considered as a designated value), while u does not. This motivates our choice as to $u < i$. \square

The implication \rightarrow is a four-valued extension of the classical implication. While its arguments range over the four logical values the truth value of the implication is t or f . The implication is used to interpret clauses of 4QL. Whenever the body of a clause has the value f or u , the truth value of the clause is defined to be t . Intuitively, this reflects our intention

not to draw conclusions from false or unknown information. Namely, a clause with unknown or false body is always satisfied, so one does not have to update its head. From inconsistent body we want to conclude that the head is also inconsistent. Thus, for the predecessor with value \mathbf{i} the implication is \mathbf{t} if the successor is \mathbf{i} , and \mathbf{f} otherwise. The implication is \mathbf{t} if the predecessor takes value \mathbf{t} and the successor is \mathbf{t} or \mathbf{i} . The latter case is needed to handle the situation when both head and its negation are to be derived on the basis of true assumptions.

REMARK 4. — Note that the classical equivalence

$$[(p \rightarrow q) \wedge (r \rightarrow q)] \equiv [(p \vee r) \rightarrow q],$$

allowing one to consider several rules as a single one, does not hold in the four-valued setting we deal with. For example, $(\mathbf{i} \rightarrow \mathbf{t}) \wedge (\mathbf{t} \rightarrow \mathbf{t})$ is \mathbf{t} , while $(\mathbf{i} \vee \mathbf{t}) \rightarrow \mathbf{t}$ is \mathbf{f} .

Observe also that the negation defined in Table 1 is not truth reversing ($\mathbf{i} > \mathbf{u}$ but not $\neg \mathbf{i} < \neg \mathbf{u}$). Our definition of \neg reflects the intuition that when we have no information as to the truth or falsity of p then we still have no such information about $\neg p$. This usually is not questioned in the three-valued case without \mathbf{i} , where $\neg \mathbf{u} \stackrel{\text{def}}{=} \mathbf{u}$. Similarly, having inconsistent p , we still have inconsistent $\neg p$. In order to have a truth reversing negation, we would have to accept that $\neg \mathbf{i} \stackrel{\text{def}}{=} \mathbf{u}$ and $\neg \mathbf{u} \stackrel{\text{def}}{=} \mathbf{i}$. We do not find this intuitive. Rather than being driven by a priori assumptions as to properties of negations and orders, our choice reflects our intuitions. \square

DEFINITION 5. — By an interpretation we mean any set of ground literals. The truth value of a literal ℓ in interpretation \mathcal{I} , denoted by $\mathcal{I}(\ell)$, is the value defined as follows:

$$\mathcal{I}(\ell) \stackrel{\text{def}}{=} \begin{cases} \mathbf{t} & \text{if } \ell \in \mathcal{I} \text{ and } (\neg \ell) \notin \mathcal{I} \\ \mathbf{i} & \text{if } \ell \in \mathcal{I} \text{ and } (\neg \ell) \in \mathcal{I} \\ \mathbf{u} & \text{if } \ell \notin \mathcal{I} \text{ and } (\neg \ell) \notin \mathcal{I} \\ \mathbf{f} & \text{if } \ell \notin \mathcal{I} \text{ and } (\neg \ell) \in \mathcal{I}. \end{cases}$$

The definition of interpretation is extended for formulas built from literals using \vee , \wedge , \neg and \rightarrow according to Table 1.

Observe that in our approach we deal with direct contradictions, but also with more general notion of inconsistency. Namely, as discussed in [CM02],

“the contradictoriness of a given theory/logic was to be identified with the fact that it derives at least some pairs of formulas of the form A and $\neg A$, while inconsistency was usually talked about as a model-theoretic property to be guaranteed so that our theories can make sense and talk about ‘real existing structures’[...]”.

Inconsistencies may appear in our approach not only due to literals contradicting each other. Heads of rules may be assigned the value \mathbf{i} by applications of rules with inconsistent bodies. In the rest of the paper we then prefer to talk about more general concept of inconsistencies rather than limited only to contradictions.

3.3. Declarative Semantics of 4QL

The declarative semantics of 4QL is defined in terms of Herbrand models. In more traditional approaches Herbrand models are intensively studied (see, e.g., [CGT89]). Our ap-

proach is based on well-supported models. Since for any set of 4QL rules there is a unique such model (see Theorem 15), it provides truth values for all formulas. Note that interpretations we deal with may contain negative literals. Therefore we assume that *Herbrand bases* consist of all (positive and negative) ground literals.

DEFINITION 6. — *A set of literals \mathcal{I} is a model of a set of rules S , denoted by $\mathcal{I} \models S$, iff for each rule $\varrho \in S$ we have that $\mathcal{I}(\text{body}(\varrho) \rightarrow \text{head}(\varrho)) = \mathbf{t}$, where it is assumed that the empty body takes the value \mathbf{t} in any interpretation.*

It should be noticed that the Herbrand base is a model of any set of rules. However, our intuition is that the knowledge represented by a set of rules should be based on the explicit knowledge represented by facts. Minimal models, if exist, may not fulfill this requirement, as shown in the following example.

EXAMPLE 7. — Let S be the following set of rules:

$$\text{wait} := \text{overloaded} \vee \text{rest_time} . \quad (2)$$

$$\text{rest_time} := \text{wait} . \quad (3)$$

$$\neg \text{overloaded} := \text{rest_time} . \quad (4)$$

$$\text{overloaded} . \quad (5)$$

Observe that:

- by (5), $\text{overloaded} \in \mathcal{I}$
- now, by (2), $\text{wait} \in \mathcal{I}$ thus, by (3), $\text{rest_time} \in \mathcal{I}$
- therefore, by (4), $\neg \text{overloaded} \in \mathcal{I}$.

Thus in every model of S , overloaded has the value \mathbf{i} .

A minimal model of S is $\mathcal{I}_{\min} = \{\text{overloaded}, \neg \text{overloaded}, \text{wait}, \text{rest_time}\}$ but the only fact of S (i.e., overloaded) has in this model value \mathbf{i} so there are no facts supporting the truth of wait and rest_time in this model. The intuitively correct model for S is

$$\mathcal{I} = \{\text{overloaded}, \neg \text{overloaded}, \text{wait}, \neg \text{wait}, \text{rest_time}, \neg \text{rest_time}\}.$$

Namely, wait obtains the value \mathbf{i} to satisfy (2). Then, to satisfy (3), rest_time obtains the value \mathbf{i} . \square

The following definitions reflect our intuitions. Note that well-supportedness closest to ours is that of [Fag94]. However, [Fag94] concerns the classical two-valued setting.

DEFINITION 8. — *Let \mathcal{I} be an interpretation and \prec be a strict partial order on \mathcal{I} . Given a set of rules S , we say that a model \mathcal{I} of S supports a rule $\varrho \in S$ w.r.t. \prec provided that:*
body(ϱ) = \emptyset or there is $\beta_j(\varrho)$ such that $\mathcal{I}(\beta_j(\varrho)) = \mathbf{t}$
and for all literals $\iota \in \beta_j(\varrho)$ we have that $\iota \prec \text{head}(\varrho)$.

DEFINITION 9. — A model \mathcal{I} of a set of rules S is well-supported provided that there exists a strict partial order \prec on \mathcal{I} such that for every literal $\ell \in \mathcal{I}$,

$$\text{– if } \mathcal{I}(\ell) = \mathbf{t} \text{ then } \mathcal{I} \text{ supports rule}(\ell) \text{ w.r.t. } \prec. \quad (6)$$

$$\text{– if } \mathcal{I}(\ell) = \mathbf{i} \text{ then (at least) one of the following conditions hold:}$$

$$\text{– } \mathcal{I} \text{ supports rule}(\ell) \text{ w.r.t. } \prec \quad (7)$$

$$\begin{aligned} &\text{– there is a rule } \varrho \in \{\text{rule}(\ell), \text{rule}(\neg\ell)\} \text{ with } \mathcal{I}(\text{body}(\varrho)) = \mathbf{i} \\ &\text{for which there is } \beta_j(\varrho) \text{ with } \mathcal{I}(\beta_j(\varrho)) = \mathbf{i} \text{ such that} \\ &\text{for all literals } \iota \in \beta_j(\varrho), \iota \prec \text{head}(\varrho). \end{aligned} \quad (8)$$

REMARK 10. — In the conditions (7)–(8) one could also expect a clause concerning $\mathcal{I}(\neg\ell) = \mathbf{i}$. On the other hand, $\mathcal{I}(\ell) = \mathbf{i}$ implies that also $\mathcal{I}(\neg\ell) = \mathbf{i}$ so the respective condition for $\mathcal{I}(\neg\ell) = \mathbf{i}$ is already included in Definition 9. For example, the interpretation $\{\text{rest}, \neg\text{rest}, \text{overloaded}, \neg\text{overloaded}\}$ is a well-supported model for the set of rules:

$$\neg \text{rest} :- \text{overloaded}. \quad (9)$$

$$\text{rest}. \quad (10)$$

$$\text{overloaded}. \quad (11)$$

$$\neg \text{overloaded}. \quad (12)$$

as well as for the set consisting of the fact

$$\neg \text{rest}.$$

together with rules (10), (11), (12). \square

EXAMPLE 11 (EXAMPLE 7 CONTINUED). — The minimal model

$$\mathcal{I}_{\min} = \{\text{overloaded}, \neg\text{overloaded}, \text{wait}, \text{rest_time}\}$$

of the set of rules considered in Example 7 is not well-supported. Namely, $\mathcal{I}_{\min}(\text{wait}) = \mathbf{t}$. According to Definition 9, there should be an order \prec such that \mathcal{I}_{\min} supports rule (2). By Definition 8,

$$\text{rest_time} \prec \text{wait}. \quad (13)$$

Since $\mathcal{I}_{\min}(\text{rest_time}) = \mathbf{t}$, a similar reasoning show that we should have

$$\text{wait} \prec \text{rest_time},$$

which together with (13) cannot hold, since \prec is required to be a strict partial order. \square

4. Properties of Well-supported Models

The following lemma explains the status of heads of rules supported by a given interpretation.

LEMMA 12. — *Let S be a set of rules and \mathcal{I} be a model of S . If \mathcal{I} supports a rule $\varrho \in S$ w.r.t. a strict partial order \prec then $\mathcal{I}(\text{head}(\varrho)) \in \{\mathbf{t}, \mathbf{i}\}$.*

PROOF — Assume that \mathcal{I} supports $\varrho \in S$ w.r.t. \prec . By Definition 8, $\text{body}(\varrho) = \emptyset$ or there is $\beta_j(\varrho)$ such that $\mathcal{I}(\beta_j(\varrho)) = \mathbf{t}$ and for all literals $\iota \in \beta_j(\varrho)$ we have that $\iota \prec \text{head}(\varrho)$.

In the first case $\text{body}(\varrho) = \emptyset$, so $[\ell :- .] \in S$. By Definition 6, the empty body \emptyset is \mathbf{t} in any interpretation, so the rule is satisfied in \mathcal{I} only when $\mathcal{I}(\text{head}(\varrho)) \in \{\mathbf{t}, \mathbf{i}\}$.

In the second case there is $\beta_j(\varrho)$ such that $\mathcal{I}(\beta_j(\varrho)) = \mathbf{t}$. Similarly to the previous case we have that ϱ is satisfied in \mathcal{I} only when $\mathcal{I}(\text{head}(\varrho)) \in \{\mathbf{t}, \mathbf{i}\}$. ■

The next two lemmas (13 and 14) are used in the proof of Theorem 15 claiming the uniqueness of well-supported models.

LEMMA 13. — *Let \mathcal{M}_1 and \mathcal{M}_2 be well-supported models for a set of formulas S . Then, for every literal ℓ , we have that $\mathcal{M}_1(\ell) = \mathbf{t}$ implies $\mathcal{M}_2(\ell) \in \{\mathbf{t}, \mathbf{i}\}$.*

PROOF — Since \mathcal{M}_1 is a well-supported model, there exists a strict partial order \prec on \mathcal{M}_1 satisfying conditions of Definition 9. Thus, for every literal $\ell \in \mathcal{M}_1$, whenever $\mathcal{M}_1(\ell) = \mathbf{t}$, there is a rule $\varrho \in S$ with $\text{head}(\varrho) = \ell$ such that \mathcal{M}_1 supports ϱ w.r.t. \prec .

We prove the lemma by induction on \prec .

- $\text{body}(\varrho) = \emptyset$ implies that the rule $[\ell :- .]$ is in S . The empty body is \mathbf{t} in any interpretation, so the rule is satisfied in \mathcal{M}_2 only when $\mathcal{M}_2(\ell) \in \{\mathbf{t}, \mathbf{i}\}$, thus the implication is trivially true
- there is $\beta_j(\varrho)$ such that $\mathcal{M}_1(\beta_j(\varrho)) = \mathbf{t}$ and for all literals ι occurring in $\beta_j(\varrho)$ we have that $\iota \prec \text{head}(\varrho)$. Since $\beta_j(\varrho)$ is a conjunction of literals, $\mathcal{M}_1(\beta_j(\varrho)) = \mathbf{t}$ implies that for all ι in $\beta_j(\varrho)$, $\mathcal{M}_1(\iota) = \mathbf{t}$. By inductive assumption, for all ι in $\beta_j(\varrho)$, $\mathcal{M}_2(\iota) \in \{\mathbf{t}, \mathbf{i}\}$, so also $\mathcal{M}_2(\beta_j(\varrho)) \in \{\mathbf{t}, \mathbf{i}\}$. Of course, \mathcal{M}_2 has to satisfy rule ϱ , so for its head ℓ we have that $\mathcal{M}_2(\ell) \in \{\mathbf{t}, \mathbf{i}\}$. ■

LEMMA 14. — *Let \mathcal{M}_1 and \mathcal{M}_2 be well-supported models for a set of formulas S . Then, for every literal ℓ , we have that $\mathcal{M}_1(\ell) = \mathbf{i}$ implies $\mathcal{M}_2(\ell) \in \{\mathbf{t}, \mathbf{i}\}$.*

PROOF — Since \mathcal{M}_1 is a well-supported model, there exists a strict partial order \prec on \mathcal{M}_1 satisfying conditions of Definition 9. We proceed by induction on \prec .

For every literal $\ell \in \mathcal{M}_1$, if $\mathcal{M}_1(\ell) = \mathbf{i}$ then:

- there is a rule ϱ with $\text{head}(\varrho) = \ell$ such that \mathcal{M}_1 supports ϱ w.r.t. \prec ; or
- there is a rule $\varrho \in S$ with $\text{head}(\varrho) \in \{\ell, \neg\ell\}$ such that there is $\beta_j(\varrho)$ satisfying $\mathcal{M}_1(\beta_j(\varrho)) = \mathbf{i}$ and for all literals ι in $\beta_j(\varrho)$, $\iota \prec \text{head}(\varrho)$.

In the first case we have that \mathcal{M}_1 supports ϱ w.r.t. \prec . If $\text{body}(\varrho) = \emptyset$ then the rule $[\ell :- .]$ is in S , so $\mathcal{M}_2(\ell) \in \{\mathbf{t}, \mathbf{i}\}$. Consider the case when there is $\beta_j(\varrho)$ such that $\mathcal{M}_1(\beta_j(\varrho)) = \mathbf{i}$ and for all literals $\iota \in \beta_j(\varrho)$ we have that $\mathcal{M}_1(\iota) = \mathbf{i}$ and $\iota \prec \text{head}(\varrho)$. Then, by inductive assumption, we obtain that for such literals $\mathcal{M}_2(\iota) \in \{\mathbf{t}, \mathbf{i}\}$. In consequence, $\mathcal{M}_2(\beta_j(\varrho)) \in \{\mathbf{t}, \mathbf{i}\}$. Since $\ell = \text{head}(\varrho)$, we have that $\mathcal{M}_2(\ell) \in \{\mathbf{t}, \mathbf{i}\}$.

It remains to consider the case when there is a rule $\varrho \in S$ with $\text{head}(\varrho) \in \{\ell, \neg\ell\}$ for which there is $\beta_j(\varrho)$ satisfying $\mathcal{M}_1(\beta_j(\varrho)) = \mathbf{i}$ and for all literals ι in $\beta_j(\varrho)$,

$\iota \prec \text{head}(\varrho)$. For all such literals, $\mathcal{M}_1(\iota) \in \{\mathbf{t}, \mathbf{i}\}$ and at least one of them takes the value \mathbf{i} . If $\mathcal{M}_1(\iota) = \mathbf{t}$ then by Lemma 13, $\mathcal{M}_2(\iota) \in \{\mathbf{t}, \mathbf{i}\}$. If $\mathcal{M}_1(\iota) = \mathbf{i}$ then by the inductive assumption $\mathcal{M}_2(\iota) \in \{\mathbf{t}, \mathbf{i}\}$. Thus $\mathcal{M}_2(\beta_j(\varrho)) \in \{\mathbf{t}, \mathbf{i}\}$, so $\mathcal{M}_2(\ell) \in \{\mathbf{t}, \mathbf{i}\}$. ■

THEOREM 15. — *For any set S of rules there is the unique well-supported model for S .*

PROOF — The fact that for every set of rules there is a well-supported model follows from Theorem 28. So here it suffices to prove that, given a set of rules S , there may not be two different well-supported models for S . Suppose the contrary, i.e., there are $\mathcal{M}_1 \neq \mathcal{M}_2$ which are well-supported models of S . This means that there is a literal, say ℓ , such that it is in one of the models and is not in the other. Without loss of generality we can assume that $\ell \in \mathcal{M}_1$ and $\ell \notin \mathcal{M}_2$.

Since $\ell \in \mathcal{M}_1$, we have that $\mathcal{M}_1(\ell) \in \{\mathbf{t}, \mathbf{i}\}$. By Lemmas 13 and 14, $\mathcal{M}_2(\ell) \in \{\mathbf{t}, \mathbf{i}\}$, so $\ell \in \mathcal{M}_2$ and a contradiction is reached. ■

5. Computing the Unique Well-Supported Model

Let us now present an algorithm for computing the unique well-supported model for a given set of rules.

DEFINITION 16. — *Let S be a set of rules.*

- By $\mathcal{L}(S)$ we denote the set of relation symbols appearing in S .
- By a duplicate of a relation symbol $\ell \in \mathcal{L}(S)$ we understand a fresh relation symbol, for simplicity denoted by ℓ' .
- By $\mathcal{L}'(S)$ we understand the set of duplicates of relation symbols of $\mathcal{L}(S)$, i.e., $\mathcal{L}'(S) = \{\ell' \mid \ell \in \mathcal{L}(S)\}$.
- By $\text{Pos}(S)$ we understand the DATALOG program obtained from S by replacing each negative literal $\neg\ell$ of S by its duplicate ℓ' .

The algorithm is shown in Figure 1. The following example illustrates its execution.

EXAMPLE 17. — To illustrate the algorithm given in Figure 1, consider set of rules discussed in Example 7 together with rules:

$$\text{good_mood} \text{ :- } \text{rested} \vee \text{success} . \quad (17)$$

$$\neg \text{rested} \text{ :- } \neg \text{rest_time} . \quad (18)$$

$$\text{rested} . \quad (19)$$

$$\text{success} . \quad (20)$$

Phase 1 gives $\mathcal{I}_1^S = \{\text{overloaded}, \neg \text{overloaded}\}$.

Phase 2 gives the following set of rules S' :

- **Input:** a set of rules S
- **Output:** the unique well-supported model \mathcal{I}^S for S .

- 1) Phase 1 (*finding basic inconsistencies*):
 - a) compute the least Herbrand model \mathcal{I}_0^S of $Pos(S)$
 - b) let $\mathcal{I}_1^S \stackrel{\text{def}}{=} \{\ell, \neg\ell \mid \ell, \ell' \in \mathcal{I}_0^S\}$
- 2) Phase 2 (*finding potentially true literals*):
 - a) let $S' = \{\varrho \mid \varrho \in S \text{ and } \mathcal{I}_1^S(\text{head}(\varrho)) \neq \mathbf{i}\}$
 - b) set \mathcal{I}_2^S to be the least Herbrand model for $Pos(S')$ with literals ℓ' substituted by $\neg\ell$
- 3) Phase 3 (*reasoning with inconsistency*):
 - a) define the following transformation Φ^S on interpretations:

$$\Phi^S(\mathcal{I}) \stackrel{\text{def}}{=} \mathcal{I} \cup \{\ell, \neg\ell \mid \text{there is a rule } [\ell :- \beta_1 \vee \dots \vee \beta_m] \in S \quad (14)$$

$$\text{such that } \exists k \in \{1, \dots, m\} [\mathcal{I}(\beta_k) = \mathbf{i}] \quad (15)$$

$$\text{and } \neg \exists n \in \{1, \dots, m\} [(\mathcal{I}_2^S - \mathcal{I})(\beta_n) = \mathbf{t}]\}. \quad (16)$$

The transformation Φ^S is monotonic (see Lemma 21).

Denote by \mathcal{I}_3^S the fixpoint of Φ^S obtained by iterating Φ^S on \mathcal{I}_1^S , i.e.,

$$\mathcal{I}_3^S = \bigcup_{i \in \omega} (\Phi^S)^i(\mathcal{I}_1^S)$$

- b) set $\mathcal{I}^S = \mathcal{I}_2^S \cup \mathcal{I}_3^S$.

Figure 1. The method of computing the well-supported model for the set of rules S .

```
wait :- overloaded ∨ rest_time .
rest_time :- wait .
good_mood :- rested ∨ success .
¬rested :- ¬rest_time .
rested .
success .
```

The resulting set \mathcal{I}_2^S is $\{success, rested, good_mood\}$.

Phase 3 gives the following iterations of Φ^S :

```
{overloaded, ¬overloaded}
{overloaded, ¬overloaded, wait, ¬wait}
{overloaded, ¬overloaded, wait, ¬wait, rest_time, ¬rest_time}
{overloaded, ¬overloaded, wait, ¬wait, rest_time, ¬rest_time,
  rested, ¬rested} – fixpoint.
```

Hence $\mathcal{I}_3^S = \{\text{overloaded}, \neg\text{overloaded}, \text{wait}, \neg\text{wait}, \text{rest_time}, \neg\text{rest_time}, \text{rested}, \neg\text{rested}\}$.

Finally, $\mathcal{I}^S = \{\text{success}, \text{good_mood}, \text{overloaded}, \neg\text{overloaded}, \text{wait}, \neg\text{wait}, \text{rest_time}, \neg\text{rest_time}, \text{rested}, \neg\text{rested}\}$,

which is the desired result. Namely, as discussed in Example 7,

$$\text{overloaded}, \neg\text{overloaded}, \text{wait}, \neg\text{wait}, \text{rest_time}, \neg\text{rest_time} \in \mathcal{I}.$$

By (20), $\text{success} \in \mathcal{I}$ so, by (17), $\text{good_mood} \in \mathcal{I}$. Finally, to satisfy (18), both $\text{rest}, \neg\text{rest} \in \mathcal{I}$. \square

In the following lemmas and theorem we assume that S is an arbitrary (finite) set of rules and use notation as in the algorithm shown in Figure 1.

LEMMA 18. — *If $\ell \in \mathcal{I}_1^S$ then the value of ℓ is \mathbf{i} in every model of S .*

PROOF — Follows directly from the fact that \mathcal{I}_0^S is the least Herbrand model for $\text{Pos}(S)$. \blacksquare

LEMMA 19. — *If for a well-supported model \mathcal{M} of S we have $\mathcal{M}(\ell) = \mathbf{t}$ then $\ell \in \mathcal{I}_2^S$.*

PROOF — According to Definition 9, if \mathcal{M} is a well-supported model then there is a strict partial order \prec such that $\mathcal{M}(\ell) = \mathbf{t}$ implies that there is a rule

$$[\ell :- b_1 \vee \dots \vee b_m] \in S$$

and $1 \leq l \leq m$ such that $b_l = b_{l1}, \dots, b_{li_l}, \mathcal{I}(b_l) = \mathbf{t}$ and for $1 \leq j \leq i_l$ we have that $b_{lj} \prec \ell$. This order shows a “computation” of S' forcing ℓ to be in \mathcal{I}_2^S , being the least Herbrand model for $\text{Pos}(S')$ (with literals ℓ' substituted by $\neg\ell'$). \blacksquare

LEMMA 20. — *For any literal ℓ , we have that $\mathcal{I}_2^S(\ell) \neq \mathbf{i}$.*

PROOF — Suppose that $\mathcal{I}_2^S(\ell) = \mathbf{i}$. Then, by the construction of S' in Phase 2(a) of the algorithm shown in Figure 1, $\mathcal{I}_1^S(\ell) = \mathbf{i}$. In such a case, again by the construction of S' , the rule making ℓ inconsistent is removed, so no rule with ℓ in its head appears in S' . Thus $\mathcal{I}_2^S(\ell) \neq \mathbf{i}$ and a contradiction is reached. \blacksquare

LEMMA 21. — *The transformation Φ^S is monotonic w.r.t. set inclusion \subseteq , i.e.,*

$$\mathcal{I} \subseteq \mathcal{J} \text{ implies } \Phi^S(\mathcal{I}) \subseteq \Phi^S(\mathcal{J}).$$

PROOF — Suppose $\mathcal{I} \subseteq \mathcal{J}$ and $\Phi^S(\mathcal{I}) \not\subseteq \Phi^S(\mathcal{J})$. Then there is a literal ℓ such that $\ell \in \Phi^S(\mathcal{I})$ and $\ell \notin \Phi^S(\mathcal{J})$. Observe that $\ell \notin \mathcal{I}$, for otherwise $\ell \in \mathcal{J}$, so also $\ell \in \Phi^S(\mathcal{J})$. Since $\ell \in \Phi^S(\mathcal{I})$ and $\ell \notin \mathcal{I}$, there is a rule $[\ell :- b_1 \vee \dots \vee b_m] \in S$ such that:

$$\exists k \in \{1, \dots, m\} [\mathcal{I}(b_k) = \mathbf{i}] \tag{21}$$

$$\neg \exists n \in \{1, \dots, m\} [(\mathcal{I}_2^S - \mathcal{I})(b_n) = \mathbf{t}]. \tag{22}$$

By (21) and the assumption that $\mathcal{I} \subseteq \mathcal{J}$, we have

$$\exists k \in \{1, \dots, m\} [\mathcal{J}(b_k) = \mathbf{i}]. \tag{23}$$

Suppose that $\exists n \in \{1, \dots, m\} [(\mathcal{I}_2^S - \mathcal{J})(b_n) = \mathbf{t}]$. By the assumption $\mathcal{I} \subseteq \mathcal{J}$ we have that $\mathcal{I}_2^S - \mathcal{J} \subseteq \mathcal{I}_2^S - \mathcal{I}$, so $(\mathcal{I}_2^S - \mathcal{J})(b_n) = \mathbf{t}$ implies that $(\mathcal{I}_2^S - \mathcal{I})(b_n) \in \{\mathbf{t}, \mathbf{i}\}$. However, by Lemma 20, for any literal ℓ , we have that $\mathcal{I}_2^S(\ell) \neq \mathbf{i}$, so $(\mathcal{I}_2^S - \mathcal{I})(b_n) \neq \mathbf{i}$ and we conclude that $(\mathcal{I}_2^S - \mathcal{I})(b_n) = \mathbf{t}$, which contradicts (22). Therefore,

$$\neg \exists n \in \{1, \dots, m\} [(\mathcal{I}_2^S - \mathcal{J})(b_n) = \mathbf{t}]. \quad (24)$$

By the definition of Φ^S and properties (23), (24) we have that $\ell, \neg\ell \in \Phi^S(\mathcal{J})$, which contradicts the assumption that $\ell \notin \Phi^S(\mathcal{J})$, i.e., that $\Phi^S(\mathcal{I}) \not\subseteq \Phi^S(\mathcal{J})$. ■

By the construction of \mathcal{I}_3^S , we have the following obvious proposition.

PROPOSITION 22. — *For any $\ell \in \mathcal{I}_3^S$ we have $\mathcal{I}_3^S(\ell) = \mathbf{i}$.*

LEMMA 23. — *The interpretation \mathcal{I}^S of S is a model for S .*

PROOF — Suppose \mathcal{I}^S is not a model. Then there is a rule $[\ell :- b_1 \vee \dots \vee b_m] \in S$ such that either

- (i) $\mathcal{I}^S(b_1 \vee \dots \vee b_m) = \mathbf{t}$ and $\mathcal{I}^S(\ell) \in \{\mathbf{u}, \mathbf{f}\}$, or
- (ii) $\mathcal{I}^S(b_1 \vee \dots \vee b_m) = \mathbf{i}$ and $\mathcal{I}^S(\ell) \in \{\mathbf{u}, \mathbf{f}, \mathbf{t}\}$.

Case (i) cannot hold since in Phase 2, the algorithm would assign \mathbf{t} to ℓ in \mathcal{I}_2^S and $\mathcal{I}_2^S \subseteq \mathcal{I}^S(\ell)$.

Case (ii) assumes that $\mathcal{I}^S(b_1 \vee \dots \vee b_m) = \mathbf{i}$ so the considered rule does not participate in Phase 2 of the algorithm. It is the only rule with ℓ in the head. Therefore, the rule cannot be assigned \mathbf{t} , so by Definition 6 and semantics of implication given in Table 1, the rule is assigned \mathbf{f} .

Thus both cases, (i) and (ii), lead to contradiction. ■

In the rest of this section we prove that the interpretations obtained by our method are well-supported models.

DEFINITION 24. — *Let S be a set of rules. We define a family $\{L_j^S \mid j \in \omega\}$ of disjoint subsets of the Herbrand base of S as follows:*

$$- L_0^S \text{ is the set of facts in } S \quad (25)$$

$$- L_{i+1}^S \text{ is the set of all literals } \ell \text{ such that rule}(\ell) \text{ has a body component } \beta_r(\text{rule}(\ell)) \text{ consisting exclusively of literals in } \bigcup_{k \leq i} L_k^S. \quad (26)$$

Note that $\text{rule}(\ell)$ considered in the above definition may have several body components $\beta_n(\text{rule}(\ell))$ satisfying (26).

To show that \mathcal{I} constructed by our algorithm is well-supported we need a strict partial order on literals of a given set S satisfying conditions of Definition 9. The required order, denoted by \prec^S , is defined below.

DEFINITION 25. — The order \prec^S is defined as the transitive closure of the following relation \prec^S :

$$\ell' \prec^S \ell \text{ iff } \ell \in L_{i+1}^S \text{ for some } i \geq 0 \text{ and } \ell' \text{ occurs in some } \beta_n(\text{rule}(\ell)) \text{ consisting exclusively of literals in } \bigcup_{k \leq i} L_k^S. \quad (27)$$

EXAMPLE 26. — Consider the Herbrand base of the set of rules in Example 7. We get $L_0^S = \{\text{overloaded}\}$, $L_1^S = \{\text{wait}\}$, $L_2^S = \{\text{rest_time}\}$, $L_3^S = \{\neg\text{overloaded}\}$ and the defined strict order is $\text{overloaded} \prec^S \text{wait} \prec^S \text{rest_time} \prec^S \neg\text{overloaded}$. Note that:

– in the case of the minimal model $\{\text{overloaded}, \neg\text{overloaded}, \text{wait}, \text{rest_time}\}$ the above order does not satisfy condition (6) of Definition 9 since in the rule (2) there should be a body component which obtains the value **t** and consisting of literals smaller w.r.t. \prec^S than wait . The only candidate is rest_time but $\text{rest_time} \not\prec^S \text{wait}$

– in the case of the intuitively correct model considered in Example 7, $\{\text{overloaded}, \neg\text{overloaded}, \text{wait}, \neg\text{wait}, \text{rest_time}, \neg\text{rest_time}\}$, the above order satisfies conditions of Definition 9 and the model is well-supported. \square

PROPOSITION 27. — The relation \prec^S (see Definition 25) is a strict partial order.

PROOF — As the sets $L_j^S(\mathcal{I})$ are disjoint, it follows by (27) that the relation \prec^S is irreflexive and asymmetric. Consequently, its transitive closure \prec^S is a strict partial order. \blacksquare

We can now formulate and prove the main result.

THEOREM 28. — For any set of rules S , its model \mathcal{I}^S constructed by algorithm given in Figure 1 is well-supported.

PROOF — By Proposition 27, \prec^S is a strict partial order. We shall show that it satisfies conditions of Definition 9 with \prec replaced by \prec^S .

We have two cases:

The case when $\mathcal{I}^S(\ell) = \mathbf{t}$.

Note that $\mathcal{I}^S(\ell) = \mathbf{t}$ only when $\ell \in \mathcal{I}_2^S - \mathcal{I}_3^S$. Since \mathcal{I}_2^S is the least Herbrand model for $\text{Pos}(S')$ with literals ℓ' substituted by $\neg\ell$, the rule $\text{rule}(\ell)$ has a body component evaluated to **t**, with all literals smaller w.r.t. \prec^S than ℓ , which has not been changed by iterating Φ in Phase 3. This shows condition (6) of Definition 9.

The case when $\mathcal{I}^S(\ell) = \mathbf{i}$.

If $\mathcal{I}^S(\ell) = \mathbf{i}$ then $\ell \in \mathcal{I}_1^S$ or $\ell \in \mathcal{I}_3^S - \mathcal{I}_1^S$.

Consider first the case when $\ell \in \mathcal{I}_1^S$. The interpretation of $\ell \in \mathcal{I}_1^S$ is obtained from the least Herbrand model of $\text{Pos}(S)$. Therefore ℓ and $\neg\ell$:

- (a) both satisfy condition (7) (when they both have been derived using rules with body components evaluated to **t**); or
- (b) both satisfy condition (8) (when they both have been derived using rules with body components evaluated to **i**); or

(c) one of them satisfies condition (7) and the other satisfies condition (8).

In the case (a) we have that ℓ satisfies (7). In cases (b) and (c), ℓ satisfies (8).

It remains to consider the case when $\ell \in \mathcal{I}_3^S - \mathcal{I}_1^S$, i.e., when ℓ has been added to \mathcal{I}_3^S by iterating Φ in Phase 3. This happens when the body of $rule(\ell)$ (or $rule(\neg\ell)$) became evaluated to **i**. There are two cases:

- $\ell \in \mathcal{I}_2^S$: a body component $\beta_i(rule(\ell))$ (or $\beta_i(rule(\neg\ell))$) has been **t** and all literals in that component have been smaller w.r.t. \prec^S than ℓ . This $\beta_i(rule(\ell))$ (or $\beta_i(rule(\neg\ell))$) satisfies the condition (7) of Definition 9; or
- $\ell \notin \mathcal{I}_2^S$: a body component $\beta_i(rule(\ell))$ (or $\beta_i(rule(\neg\ell))$) has been **i** and all literals in that component have been smaller w.r.t. \prec^S than ℓ . This $\beta_i(rule(\ell))$ (or $\beta_i(rule(\neg\ell))$) satisfies the condition (8) of Definition 9. ■

REMARK 29. — Observe that Theorem 28 provides a PTIME method for verifying whether a given model is well-supported since \prec^S can be constructed in deterministic polynomial time. Definition 9 does not provide a direct PTIME method due to existential quantification over \prec^S . □

6. Layered Architecture

In this section we introduce and discuss external literals allowing us to express non-monotonic rules. The idea is similar to stratification but the problem is not with negation but with external literals. In this section we use well-known techniques related to stratified DATALOG⁺. All necessary definitions and theorems related to stratifications can be found, e.g., in [AHV96]. External literals have also been considered in [VMS09].

The architecture discussed in this section is equivalent to the modular architecture of [MS11]. However, it simplifies the proof of Lemma 37 and also provides intuitions related to well-known stratified programs.

Let \mathbb{M}, \mathbb{R} be disjoint sets. In what follows we assume that all relation symbols are of the form $M.R$, where $M \in \mathbb{M}$ and $R \in \mathbb{R}$. Intuitively, members of \mathbb{M} are names of “modules” or “services” and members of \mathbb{R} are classical relation symbols.

DEFINITION 30. — An external literal is an expression of one of the forms:

$$M.R, \neg M.R, M.R \text{ IN } T, \neg M.R \text{ IN } T,$$

where:

- $M \in \mathbb{M}$ is called the reference module of the external literal;
- $T \subseteq \{\mathbf{t}, \mathbf{f}, \mathbf{i}, \mathbf{u}\}$ (if $T = \emptyset$ then $\ell \text{ IN } T$ is **f**).

We write $\ell = v$ to stand for $\ell \text{ IN } \{v\}$. For an extended literal ℓ , by $module(\ell)$ we denote its reference module.

The literal $\neg M.R \text{ IN } T$ is to be read as “ $(\neg M.R) \text{ IN } T$ ” rather than “ $\neg(M.R \text{ IN } T)$ ”.

DEFINITION 31. — By an extended rule we understand a rule ϱ of the form (1), where relation symbols are replaced by external literals assuming that $head(\varrho)$ is of the form $M.R$ or $\neg M.R$.

The following example shows a possible use of external literals.

EXAMPLE 32. — The following rules locally close *loc*, where *loc*(*X*, *Y*, *T*) means that object *X* has location *Y* at timepoint *T*:

$$\begin{aligned}
 K.loc(X, Y, T) :- & \quad L.nextTime(T, S), & \quad - T \text{ is the timepoint next to } S \\
 & \quad L.house(X), & \quad - X \text{ is a house} \\
 & \quad L.loc(X, Y, S), & \quad - \text{location of } X \text{ at time } S \text{ is } Y \\
 & \quad L.chLoc(X, S) \text{ IN } \{u, f\}. & \quad - \text{location change of } X \text{ is } u \text{ or } f.
 \end{aligned}$$

Intuitively, the above rule states that houses do not change their location no matter whether *M*'s database contains information as to the change of location or not. \square

DEFINITION 33. — A set of rules *S* is well-layered iff there is a mapping $\kappa_S : \mathbb{M} \rightarrow \omega$ such that for every rule $\varrho \in S$,

- if ℓ is a literal of the form *M.R* or $\neg M.R$ appearing in *body*(ϱ),

$$\kappa_S(\text{module}(\text{head}(\varrho))) \geq \kappa_S(\text{module}(\ell))$$

- if ℓ is a literal of the form *M.R* IN *T*, $\neg M.R$ IN *T* appearing in *body*(ϱ),

$$\kappa_S(\text{module}(\text{head}(\varrho))) > \kappa_S(\text{module}(\ell)).$$

By an immediate adaptation of the algorithm for checking stratifiability of a set of rules of *DATALOG*[⊃] (see, e.g., [AHV96]), we have the following proposition.

PROPOSITION 34. — Checking whether a set of rules *S* is well-layered takes time polynomial in the size of *S*.

We now have the following definition.

DEFINITION 35. — An extended 4QL program is any finite well-layered set of extended rules.

7. Complexity Issues

7.1. Expressing Stratified *Datalog*[⊃]

Let *P* be a stratified *DATALOG*[⊃] program. Let $\{1, 2, \dots, n\}$ be all strata of *P*. First, we replace all rules with the same head ℓ as a single rule whose body is the disjunction of all bodies of rules with head ℓ . This replacement is correct, since here we deal with the standard two-valued semantics for stratified *DATALOG*[⊃] programs.

For each stratum $i = 1, \dots, n$ of *P* we take distinct symbols $M_i, N_i \in \mathbb{M}$ and replace the stratum by:

- rules obtained from rules appearing in *i*-th stratum of *P* by replacing each relation symbol *R* by:
 - $M_i.R$ if *R* is defined in stratum *i*
 - $N_j.R$ if *R* is defined in stratum *j* with $j < i$

– for each relation R defined in stratum i we add the following rules, closing the relation R as the Closed World Assumption CWA does:

$$\begin{aligned} N_i.R &:- M_i.R = \mathbf{t}. \\ \neg N_i.R &:- M_i.R \text{ IN } \{\mathbf{f}, \mathbf{u}\}. \end{aligned}$$

The result of p appearing in i -th stratum of P is given by $N_i(p)$.

Observe that the resulting set of rules is an extended 4QL program which can never lead to inconsistency.

EXAMPLE 36. — Consider a stratified program, consisting of three strata, shown in the first column of Table 2. Using the method described above we obtain the equivalent 4QL program shown in the second column of Table 2, where:

- $N_1.r$ provides the value of r
- $N_2.s, N_2.q$ provide values respectively for s and q
- $N_3.p$ provides the value of p .

□

Table 2. A stratified program and its 4QL representation.

$\left. \begin{array}{ll} p :- \neg q. & - \text{stratum 3} \\ p :- r. & - \text{stratum 3} \\ \hline q :- r. & - \text{stratum 2} \\ s :- q. & - \text{stratum 2} \\ \hline r. & - \text{stratum 1} \end{array} \right\} \Rightarrow$	$N_3.p :- M_3.p = \mathbf{t}. \quad - \text{CWA}(p)$
	$\neg N_3.p :- M_3.p \text{ IN } \{\mathbf{f}, \mathbf{u}\}. \quad - \text{CWA}(p)$
	$M_3.p :- \neg N_2.q \vee N_1.r. \quad - \text{layer 3}$
	$N_2.q :- M_2.q = \mathbf{t}. \quad - \text{CWA}(q)$
	$\neg N_2.q :- M_2.q \text{ IN } \{\mathbf{f}, \mathbf{u}\}. \quad - \text{CWA}(q)$
	$N_2.s :- M_2.s = \mathbf{t}. \quad - \text{CWA}(s)$
	$\neg N_2.s :- M_2.s \text{ IN } \{\mathbf{f}, \mathbf{u}\}. \quad - \text{CWA}(s)$
	$M_2.q :- N_1.r. \quad - \text{layer 2}$
	$M_2.s :- M_2.q. \quad - \text{layer 2}$
	$N_1.r :- M_1.r = \mathbf{t}. \quad - \text{CWA}(r)$
	$\neg N_1.r :- M_1.r \text{ IN } \{\mathbf{f}, \mathbf{u}\}. \quad - \text{CWA}(r)$
	$M_1.r. \quad - \text{layer 1}$

Due to the above construction we have the following lemma.

LEMMA 37. — Every stratified DATALOG[−] program can be expressed by an extended 4QL program.

7.2. Complexity of Layered 4QL

First observe that Algorithm provided in Figure 1 involves standard DATALOG computations in Phase 1 and Phase 2 and a fixpoint computation in Phase 3. Such computations have PTIME complexity (see, e.g., [AHV96]). Moreover, extended 4QL programs can be constructed layer by layer, starting from the lowest layer. Therefore we have the following theorem.

THEOREM 38. — 4QL with modules has PTIME data complexity.

Since stratified DATALOG captures PTIME on ordered structures (see, e.g., Theorem 15.4.8 in [AHV96]), by Lemma 37 we have that extended 4QL captures PTIME queries.

THEOREM 39. — *Extended 4QL captures PTIME queries on ordered structures.*

8. Conclusions

In the paper we have investigated the query language 4QL originally outlined in [MS11]. We defined a semantics based on the notion of well-supported model and we proposed an algorithm for computing well-supported models. The language is simple yet powerful.

We focussed on logical foundations and complexity of 4QL, proving the correctness of the algorithm for computing well-supported models and showing that 4QL has PTIME data complexity and captures PTIME.

In summary,

- 4QL provides a very flexible mechanism for dealing with the lack of knowledge and resolving possible inconsistencies in an application dependent manner
- 4QL is powerful enough to express large classes of nonmonotonic rules known from the literature (see [MS11])
- 4QL can be used as a rule language for Semantic Web and robotics applications.

There are still interesting questions concerning 4QL. Apart from implementing and applying 4QL, there are still many theoretical issues. Perhaps the most important are:

- provide a top-down method for query answering running in deterministic polynomial time
- extend the method to the case of infinite domains
- provide optimization techniques improving the performance of query answering, not necessarily by computing the whole well-supported model
- provide techniques for computing the well-supported model after a database update, on the basis of the model computed before the update.

Acknowledgments

We would like to thank Wlodek Drabent and two anonymous reviewers for critical reading of this paper and supplying many helpful comments and remarks.

This work has been supported in part by the Polish MNiSW grant N N206 399334.

9. References

- [ADP05] J. Alcântara, C.V. Damásio, and L.M. Pereira. An encompassing framework for paraconsistent logic programs. *J. Applied Logic*, 3(1):67–95, 2005.
- [AD98] S. Abiteboul and O.M. Duschka. Complexity of Answering Queries Using Materialized Views. *Proceedings PODS*, 254–263, 1998.

- [AHV96] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley Pub. Co., 1996.
- [Ari02] O. Arieli. Paraconsistent declarative semantics for extended logic programs. *Ann. Math. Artif. Intell.*, 36(4):381–417, 2002.
- [BCG07] J.-Y. Béziau, W. Carnielli, and D.M. Gabbay, editors. *Handbook of Paraconsistency*, volume 9 of *Logic and cognitive systems*. College Publications, 2007.
- [Bel77] N.D. Belnap. A useful four-valued logic. In G. Eptein and J.M. Dunn, editors, *Modern Uses of Many Valued Logic*, pages 8–37. Reidel, 1977.
- [CGT89] S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about Datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1:146–166, 1989.
- [CM02] W. Carnielli and J. Marcos. A taxonomy of c-systems. In W. Carnielli, M.E. Coniglio, and I.M.L. D’Ottaviano, editors, *Paraconsistency - the Logical Way to the Inconsistent*, volume 228 of *Lecture Notes in Pure and Applied Mathematics*, pages 1–94, 2002.
- [CMdA00] W.A. Carnielli, J. Marcos, and S. de Amo. Formal inconsistency and evolutionary databases. *Logic and Logical Philosophy*, 8:115–152, 2000.
- [CMM07] W. Carnielli, E.C. Marcelo, and J. Marcos. Logics of formal inconsistency. In Béziau et al. [BCG07], pages 848–852.
- [dACM02] S. de Amo, W.A. Carnielli, and J. Marcos. A logical framework for integrating inconsistent information in multiple databases. In *FoIKS ’02: Proc. of the 2nd Int. Symposium on Foundations of Information and Knowledge Systems*, pages 67–84. Springer-Verlag, 2002.
- [dAP07] S. de Amo and M.S. Pais. A paraconsistent logic approach for querying inconsistent databases. *International Journal of Approximate Reasoning*, 46:366–386, 2007.
- [DŁSS06] P. Doherty, W. Łukaszewicz, A. Skowron, and A. Szałas. *Knowledge representation techniques. A rough set approach*, volume 202 of *Studies in Fuziness and Soft Computing*. Springer, 2006.
- [DMS06] P. Doherty, M. Magnusson, and A. Szałas. Approximate databases: A support tool for approximate reasoning. *Journal of Applied Non-Classical Logics*, 16(1-2):87–118, 2006. Special issue on Implementation of logics.
- [DP98] C.V. Damásio and L.M. Pereira. A survey of paraconsistent semantics for logic programs. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, pages 241–320, 1998.
- [Dub08] D. Dubois. On ignorance and contradiction considered as truth-values. *Logic Journal of the IGPL*, 16(2):195–216, 2008.
- [Fag94] F. Fages. Consistency of Clark’s completion and existence of stable models. *Methods of Logic in Computer Science*, 1:51–60, 1994.
- [Fit85] M.C. Fitting. A Kripke-Kleene semantics for logic programs. *J. Log. Program.*, 2(4), 1985.
- [Fit02] M.C. Fitting. Fixpoint semantics for logic programming. a survey. *Theoretical Computer Science*, 278(1-2):25–51, 2002.
- [GL91] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3/4):365–386, 1991.
- [Mai10] F. Maier. Extending paraconsistent SROIQ. In *Proceedings of the Fourth Int. Conf. on Web reasoning And Rule Systems RR2010*, Lecture Notes in Computer Science. Springer, 2010. to appear.

- [MH09] Y. Ma and P. Hitzler. Paraconsistent reasoning for OWL 2. In Axel Polleres and Terrance Swift, editors, *RR*, volume 5837 of *Lecture Notes in Computer Science*, pages 197–211. Springer, 2009.
- [MS96] C. Meghini and U. Straccia. A relevance terminological logic for information retrieval. In *Proc. of SIGIR'96*, pages 197–205. ACM, 1996.
- [MS11] J. Małuszyński and A. Szałas. Living with inconsistency and taming nonmonotonicity. In G. Gottlob, editor, *Datalog 2.0*. Springer-Verlag, 2011.
- [MSV08] J. Małuszyński, A. Szałas, and A. Vitória. Paraconsistent logic programs with four-valued rough sets. In C.-C. Chan, J. Grzymala-Busse, and W. Ziarko, editors, *Proceedings of 6th International Conference on Rough Sets and Current Trends in Computing (RSCTC 2008)*, volume 5306 of *LNAI*, pages 41–51, 2008.
- [NS10] A.L. Nguyen and A. Szałas. Three-valued paraconsistent reasoning for Semantic Web agents. In P. Jędrzejowicz, N.T. Nguyen, R.J. Howlet, and L.C. Jain, editors, *4th International KES Symposium on Agents and Multi-Agent Systems*, volume 6070 of *LNAI*, pages 152–162, 2010.
- [Nut94] D. Nute. Defeasible logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 353–395, 1994.
- [OW08] S.P. Odintsov and H. Wansing. Inconsistency-tolerant description logic. part II: A tableau algorithm for $CACL^C$. *Journal of Applied Logic*, 6(3):343–360, 2008.
- [Rei78] R. Reiter. On Closed World Data Bases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, Plenum Press, 55–76, 1978.
- [SI95] C. Sakama and K. Inoue. Paraconsistent stable semantics for extended disjunctive programs. *J. Log. Comput.*, 5(3):265–285, 1995.
- [Str97] U. Straccia. A sequent calculus for reasoning in four-valued description logics. In D. Galmiche, editor, *Proc. of TABLEAUX'97*, volume 1227 of *LNCS*, pages 343–357. Springer, 1997.
- [VMS09] A. Vitória, J. Małuszyński, and A. Szałas. Modeling and reasoning in paraconsistent rough sets. *Fundamenta Informaticae*, 97(4):405–438, 2009.